N90-25536

A comparison of two neural network schemes for navigation.

Paul Munro
Department of Information Science
University of Pittsburgh
Pittsburgh PA 15260

munro@idis.lis.pittsburgh.edu

Abstract. Neural networks have been applied to tasks in several areas of artificial intelligence, including vision, speech, and language. Relatively little work has been done in the area of problem solving. Two approaches to path-finding are presented, both using neural network techniques. Both techniques require a training period. Training under the back propagation (BPL) method was accomplished by presenting representations of [current position, goal position] pairs as input and appropriate actions as output. The Hebbian/interactive activation (HIA) method uses the Hebbian rule to associate points that are nearby. A path to a goal is found by activating a representation of the *goal* in the network and processing until the current position is activated above some threshold level. BPL, using back-propagation learning, failed to learn, except in a very trivial fashion, that is equivalent to table lookup techniques. HIA, performed much better, and required storage of fewer weights. In drawing a comparison, it is important to note that back propagation techniques depend critically upon the forms of representation used, and can be sensitive to parameters in the simulations; hence the BPL technique may yet yield strong results.

Introduction

Description of the problem.

A *map* is given, which is a represention of landmarks and allowed paths and/or obstacles in the relevant region of space. Given an arbitrary pair of points, **I** (initial) and **G** (goal), the problem is to compute a sequence of actions which will bring the subject from **I** to **G**. Several approaches to this classic problem have been put forward (see for example, Brooks, 1983). These tend to rely upon explicit geometrical computations on polygonal representations of obstacles. In contrast, any geometrical considerations in the neural network approaches described below are *implicit*; that is, they are emergent artifacts of the learning processes. Two principles for processing and training of neural networks are briefly described in this section. More detailed treatments can be found in the references.

Back-propagation learning

Back propagation (Rumelhart, Hinton, & Williams, 1986) is a general algorithmic framework for training a feed-forward network of semi-linear units by randomly selecting pairs of input-output patterns from a training set and incrementally adjusting the network parameters, such that the network produces the appropriate output for a given input. The parameters of the network are usually, but not necessarily, restricted to the weights on the links (edges, in graph theoretic terminology) between the units (nodes). Initially, the parameters are set to random values. With each presentation of an input-output pair, the network produces a response to the input, which is compared to the desired output; the back-propagation learning (BPL) algorithm specifies a method for adjusting the network parameters, such that the discrepancy between the response and the desired output is reduced. The procedure is based on a gradient descent of the parmeter vector across an error measure. Like other gradient descent techniques, BPL is not guaranteed to find the global minimum; instead,

PRECEDING PAGE BLANK NOT FILMED

it often gets stuck in local minima, which may nevertheless result in acceptable performance by the network. As originally conceived, BPL was limited to static patterns; however there has been recent progress in processing time-varying inputs. (e.g. Jordan, 1987; Elman, 1988).

Hebbian/Interacive Activation

An interactive activation network (McClelland & Rumelhart, 1981) consists of a population of "neuron-like" elements, each representing an identifiable concept, in most implementations. The nodes are connected with positive weights, if their concepts are positively associated, and with negative weights if they are negatively associated. Normally, the weights are "hard-wired"; that is, the weights are preset and do not modify. However Hebb's postulate (1949) can be realized as a differential equation for learning in such networks, as has been done in other models, such as the Brain State in the Box (BSB) model of Anderson (1977).

A BPL approach to navigation: Method and results

The back propagation algorithm is typically applied to categorization problems, by learning an input-output mapping, where the inputs are exemplars and the outputs are categories. Jordan (1987) showed how a network could be trained to learn sequences, by partitioning the input into a represention labelling the sequence and a representation of one element of the sequence, and the output as a representation of the successor element in the sequence. Below, a similar scheme is applied to the navigation problem. The input is partitioned into a representation of the current state and a representation of the desired (goal) state. The output drives some sort of effector which changes the current state. The network architecture is shown in Figure 1. The current state and goal states are represented as patterns of activation across sets of input units. Each of the input units is connected to each of the "hidden" units (hidden, because they do not interact with the environment external to the network) in the next layer. Each of the hidden units is connected to each of the output units. The connection matrices are symbolized by the bold arrows in Figure 1.

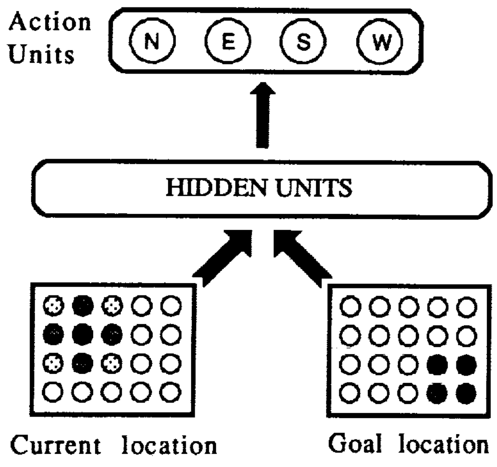

Figure 1. Basic route-finding network. Input units specify current and goal locations. The network generates an appropriate action as the output. The current location is updated according to the interaction of the action with the physics of the environment.

Attempts were made to train such a network on a very simple environment, consisting of a 5 by 5 grid of cells, each accessable by a single step from its 4 (N, E, S, and W) neighbors. With a small set of training data, the network was able to learn the steps in that set perfectly. However, if the set became too large, performance would suffer. The performance of the network (with the addition of a second layer of hidden units) is shown in Figure 2. For each of the 25 possible goals, a five by five matrix of arrows depicts the motion taken by the network . A circle indicates where the position is identical to the goal; thus for each matrix the circle is the target. Note that while the trend is generally correct, the network makes errors that lead to dead ends (edges) or limitless oscillations (for example, when two arrows point toward each other).

```
O^<<^     <O<<^     <^O>^     >^>O^     >>>>O
><<<<     >^<<<     >>^<<     >>>^^     >>>>^
^^<<<     ^^^<<     >^^^<     >>^^^     >>>^^
^^^<<     ^^^^<     ^^^^^     >^^^^     >>^^^
^^^^^     ^^^^^     ^^^^^     >^^^^     >>^^^

<<<<<     <<<<^     v^v<^     >>>>^     >>>>^
O<<<<     >O<<<     >>O<<     >>>O^     >>>>O
^<<<<     >^<<<     >>^<<     >>>^<     >>>><
^^<<<     ^^^<<     >^^^<     >>^^^     >>>^^
<^^<<     ^^^^^     >^^^^     >>^^^     >>^^>

<<<<<     v<vv<     v>vv^     v>vv^     >>>>^
<<<<<     vv<<<     >>v<<     >>>v<     >>>>>
O<<<<     >O<<<     >>O<<     >>>O<     >>>>O
<<<<<     ^^<<<     >>^<<     >>>^^     >>>>^
<<<<<     v^^<^     v>^^^     >>^^^     >>^^>

v<<v<     vvvv<     vvvv<     v>vvv     v>vv>
vv<<<     vvv<<     >vvv<     >>vvv     >>>vv
v<<<<     >v<<<     >>v<<     >>>v<     >>>>v
O<<<<     >O<<<     >>O<<     >>>O<     >>>>O
<<<<<     v<<<<     >>^^>     v>^v>     >>>v>

vvvv<     vvvv<     vvvvv     vvvvv     v>vvv
vvv<<     vvvv<     vvvvv     >vvvv     >>vvv
vv<<<     vvv<<     >vvv<     >>vvv     >>>vv
<<<<<     vv<<<     >>v<<     >>>v<     >>>>v
O<<<<     vO<<v     v>O>v     >>>O>     v>>vO
```

Figure 2. The results from a fully connected BPL network. See text for description.

It was generally found that learning was much better when the patterns were presented to the network independently of the previous pattern. In the initial investigations, a particular goal was held constant while the network was trained on a sequence of steps leading to the goal, after which the goal was shifted to a random location. However, under such a training schedule, the goal position can remain constant too long, such that the weights from the current position representation "forget" what they learned with respect to other goals.

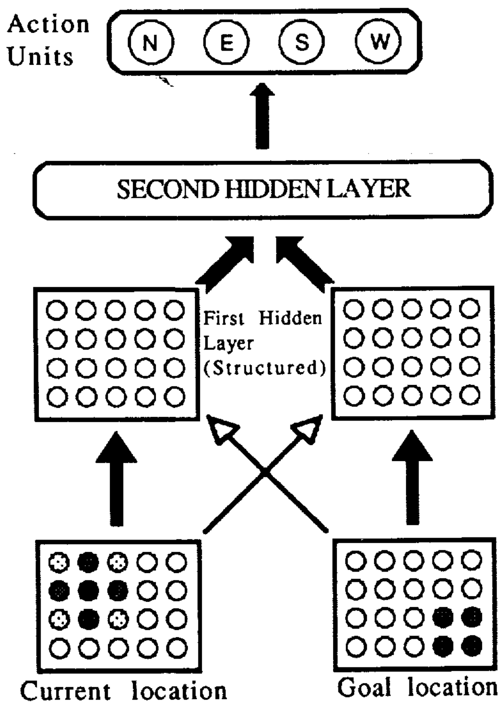

Figure 3. In this network, two sets of units have been inserted between the input layer and the hidden unit layer. One has a set of one-to-one connections with the GOAL input units and is fully connected to the CURRENT input units, and the other is connected in a complementary fashion. This "structured" hidden layer facilitates learning, but leads to poor generalization.

To remedy this, a more complex architecture was introduced (see Figure 3), by inserting another layer of hidden units into the previous structure between the input layer and the hidden layer. This new hidden layer consists of two sets of units. One set has one unit corresponding to each unit in the goal location input layer and receives input from that unit alone among the goal location input units; all of the units in that set receive input from *all* units in the current location input set. That is, that set in the first hidden layer receives one-to-one connections (thin arrow) from the goal location input set and is fully connected to the current location input set (thick arrow). The other hidden set has a complementary set of connections. The one-to-one connections were *gates*, or *multiplicative connections*; that is, unless input was received from one of these connections the hidden unit did not respond.

With this architecture, the network was able to learn the 5 by 5 environment perfectly, as shown in Figure 4. However, in this case learning is quite brittle. The network is now nothing more than a lookup table, since it has specific weights corresponding to every input combination. Thus, there is no generalization of information from one learning trial to any other situation.

```
O<<<<     >O<<<     >>O<<     >>>O<     >>>>O
^<<<<     >^<<<     >>^<<     >>>^<     >>>>^
^^<<<     ^^^<<     >^^^<     >>^^^     >>>^^
^^^<<     ^^^^<     ^^^^^     >^^^^     >>^^^
^^^^<     ^^^^^     ^^^^^     ^^^^^     >^^^^

v<<<<     >v<<<     >>v<<     >>>v<     >>>>v
O<<<<     >O<<<     >>O<<     >>>O<     >>>>O
^<<<<     >^<<<     >>^<<     >>>^<     >>>>^
^^<<<     ^^^<<     >^^^<     >>^^^     >>>^^
^^^<<     ^^^^<     ^^^^^     >^^^^     >>^^^

vv<<<     vvv<<     >vvv<     >>vvv     >>>vv
v<<<<     >v<<<     >>v<<     >>>v<     >>>>v
O<<<<     >O<<<     >>O<<     >>>O<     >>>>O
^<<<<     >^<<<     >>^<<     >>>^<     >>>>^
^^<<<     ^^^<<     >^^^<     >>^^^     >>>^^

vvv<<     vvvv<     vvvvv     >vvvv     >>vvv
vv<<<     vvv<<     >vvv<     >>vvv     >>>vv
v<<<<     >v<<<     >>v<<     >>>v<     >>>>v
O<<<<     >O<<<     >>O<<     >>>O<     >>>>O
^<<<<     >^<<<     >>^<<     >>>^<     >>>>^

vvvv<     vvvvv     vvvvv     vvvvv     >vvvv
vvv<<     vvvv<     vvvvv     >vvvv     >>vvv
vv<<<     vvv<<     >vvv<     >>vvv     >>>vv
v<<<<     >v<<<     >>v<<     >>>v<     >>>>v
O<<<<     >O<<<     >>O<<     >>>O<     >>>>O
```

Figure 4. The results from a partially connected BPL network. See text for description.

A Hebbian/Interactive activation approach to navigation: Method and results

In this network, the environment was represented similarly to the above input representation, in that there is a unit for every landmark in the environment. Again, in consideration of designing the computer simulation, a rectangular grid was used. However, the architecture was quite different. In this network, all units were connected (initially) to all other units. Simulations using this model were performed in more complex environments; here, not every grid element was connected to its four neighbors. Instead environments, such as that shown in Figure 5, were used.

```
X -- X -- X    X -- X -- X
|         |    |         |
X    X -- X    X    X -- X
|    |    |    |         |
X -- X    X -- X -- X -- X
          |
X -- X -- X -- X -- X -- X
|         |    |
X    X -- X    X -- X    X
|         |    |         |
X -- X -- X -- X -- X -- X
```

Figure 5. An example of a maze environment, such as was used in the HIA simulations.

Training was accomplished by an exploratory "wandering" process through the maze. Each cycle of the simulation began with taking a random step from the current position to a neighbor (neighbors in this case were defined by the links in the maze), with no backtracking unless necessary. Upon arrival at a node, the corresponding unit in the activation network was activated. Activatation in each unit would decay by a fraction α, with each cycle of the simulation. The weights bewteen units would increase in proportion to the product of the activity in the two units, and decay by another factor β. Appropriate choice of α and β led to a situation in which the weights between adjacent units were much stronger than the weights bewteen units two or more steps removed. Hence, these were all set to zero, and the neural net became isomorphic to the maze. This network was used to compute paths between arbitrary points in the maze by the following three stage procedure:

[1] The unit, *g*, corresponding to the goal is stimulated continuously at a high level, K, and activation spreads through the network via repeated iteration of matrix multiplication and simultaneous exponential decay, until the unit, *c*, corresponding to the current position is activated to a criterion level θ:

$$A_j(t) = \sum_k W_{jk} A_k(t) - \eta A_j(t) \quad \text{for } j \neq g$$

$$A_g(t) = K \text{ until } A_c > \theta$$

[2] The resulting pattern of activation **A** is then multiplied, element by element, by the pattern of the squares of the weights connecting *c* to the other units, $\mathbf{W_c}$.

[3] The current position is then updated, by moving to the unit with the greatest resulting product:

$$c(t+1) = \text{index j that gives a maximum for } W^2_{c(t)j} A_j$$

Steps [2] and [3] are repeated until the goal is reached.

This method was found to work quite well over a set of different mazes, usually finding the shortest path. In cases where the shortest path was not found, the result was close to the optimum.

Discussion

While BPL was found to be inadequate for solving relatively simple problems, it should be recognized that it frequently requires considerable time and effort (and educated guesswork) to apply it successfully to a particular problem. The pattern representations must be carefully considered. Also, the network architecture and even such parameters as the learning rate and the momemtum (see Rumelhart, Hinton, and Williams, 1986 for a detailed description) can be critical in determining the success of a particular simulation. Thus, while the results reported above are discouraging, it is too soon to dismiss this approach.

The second technique, HIA, performed much better. The algorithm for finding a path is not especially novel; it is essentially equivalent to searching though a graph for the shortest path. The novelty is in the Hebbian modification technique used to construct the graph via temporally correlated activitations. This is somewhat sensitive to the parameters α and β. Further work is required for a general solution using this approach.

Other future plans include using more sophisticated representations for location, using multiple maps of the environment, such as maps for various types of transportation (e.g., walking vs. driving), or maps covering various scales (e.g., city maps vs. world maps). Recent work (Munro & Hirtle, 1989) has shown how the interactive activation model can account for a variety of documented psychological data, which indicates interactions between internal representations of different maps in free recall of geographical information. Conceivably, a hybrid technique, involving both the BPL and HIA methods will be used. Such a combination would probably use HIA for the high level planning and BPL to issue the action commands to the drive mechanism.

References

Anderson, J. A., Silverstein, J. W., Ritz, S. A., and Jones, R. S. (1977) Distinctive features, categorical perception, and probability learning: Some applications of a neural model. *Psychological Review,* **84**:413-451.

Brooks, R. (1983) Solving the find-path problem by good representation of free space. *IEEE Transactions on Systems, Man, and Cybernetics,* SMC **13:** 190-197.

Elman, J. L. (1988) Finding structure in time. CRL TR 8801. Center for Research in Language. University of California at San Diego.

Hebb, D. O. (1949) *The Organization of Behavior,* New York: Wiley.

Jordan, M. I. (1987) Attractor dynamics and parallelism in a connectionist sequential machine. *Proceedings of the Eighth Annual Conference of the Cognitive Science Society,* 531-546.

McClelland, J. L. and Rumelhart, D. E. (1981) An interactive activation model of context effects in letter perception: Part I. An account of basic findings. *Psychological Review,* **88**:375-407.

Munro, P. and Hirtle, S. C. (1989) An interactive activation model for priming of geographical information. *Proceedings of the Eleventh Annual Conference of the Cognitive Science Society,* [Accepted for presentation]

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986) Learning internal representations by error propagation. In: *Parallel distributed processing: Explorations in the microstructure of cognition.* D. E. Rumelhart and J. L. McClelland, eds. Cambridge, MA: MIT